

Practical Learning of Deep Gaussian Processes via Random Fourier Features

Kurt Cutajar¹Edwin V. Bonilla²Pietro Michiardi¹Maurizio Filippone¹¹Department of Data Science, EURECOM, France²School of Computer Science and Engineering, University of New South Wales, Australia

Abstract

The composition of multiple Gaussian Processes as a Deep Gaussian Process (DGP) enables a deep probabilistic approach to flexibly quantify uncertainty and carry out model selection in various learning scenarios. In this work, we introduce a novel formulation of DGPs based on random Fourier features that we train using stochastic variational inference. Our proposal yields an efficient way of training DGP architectures without compromising on predictive performance. Through a series of experiments, we illustrate how our model compares favorably to other state-of-the-art inference methods for DGPs for both regression and classification tasks. We also demonstrate how an asynchronous implementation of stochastic gradient optimization can exploit the computational power of distributed systems for large-scale DGP learning.

1 INTRODUCTION

Given their impressive performance on machine learning and pattern recognition tasks, Deep Neural Networks (DNNs) have recently attracted a considerable deal of attention in several applied domains such as computer vision and natural language processing; see, e.g., LeCun et al. (2015) and references therein.

Deep Gaussian Processes (DGPs; Damianou and Lawrence, 2013) are deep probabilistic models where the mapping between inputs and labels is constructed by composing functions specified in a nonparametric fashion. From a generative perspective, DGPs transform the inputs using a cascade of Gaussian Processes (GPs; Rasmussen and Williams, 2006) such that the output of each layer of GPs forms the input to the GPs at the next layer. An attractive feature of DGPs is

that, unlike DNNs, their probabilistic formulation allows for a principled way to tackle quantification of uncertainty and model selection, which is desirable when employing deep models in practice.

The flexibility and expressive power of DGPs, however, comes at the expense of having to deal with a model for which inference and predictions are inherently difficult. The reason is that the nonlinearity introduced by each layer makes it difficult to tractably propagate the uncertainty throughout the layers. An additional complication stems from the strong coupling of the GPs at all layers, which are further parameterized by covariance parameters to be optimized or inferred.

In this paper, we develop a practical learning framework for DGP models. In particular, our proposal introduces two sources of approximation. The first is a model approximation, whereby the GPs at all layers are approximated using random Fourier features (Rahimi and Recht, 2008; Lázaro-Gredilla et al., 2010). The second approximation relies upon stochastic variational inference to retain a probabilistic treatment of the approximate DGP model.

The appeal of our proposal is that the model approximation can be tuned by selecting an appropriate number of random Fourier features, and this is dictated by constraints on running time or hardware. The impact of the variational approximation, as opposed to characterizing the full posterior distribution over all model parameters, is the most difficult to assess. We illustrate the quality of our approximation by comparing the variational approximation of the proposed model with the inference resulting from the application of Markov chain Monte Carlo (MCMC) methods that we develop for two-layer DGP regression models.

Although our approach is amenable to any stationary covariance function, we develop it in detail for the squared exponential covariance function, which leads to a truncated trigonometric expansion. As a result, the approximate DGP model becomes a DNN with trigonometric activation functions and low-rank

weight matrices. By virtue of the connection to DNNs, we are able to employ scalable optimization and inference techniques that have been developed in the literature to train DNNs, allowing the proposed DGP model to scale to large data in the same way as DNNs.

In order to retain a probabilistic treatment of the model when scaling to large datasets, we employ stochastic variational inference techniques. In particular, we adapt the work on variational inference for DNNs (Kingma and Welling, 2014) using mini-batch-based stochastic gradient optimization for the proposed DGP model. We implement the model in TensorFlow (Abadi et al., 2015), which allows us to rely on automatic differentiation during the optimization procedure.

We demonstrate the effectiveness of our proposal in a variety of regression and classification problems by comparing it with other state-of-the-art approaches to infer DGPs (Damianou and Lawrence, 2013; Dai et al., 2016; Bui et al., 2016). The results indicate that for a given DGP architecture, our proposal is consistently faster at achieving lower errors compared to the competitors. We also obtain impressive results when applying the model to the MNIST and MNIST-8M digit classification problems, the latter of which contains over 8 million high-dimensional observations. Such an undertaking would not have been computationally feasible using a regular GP model, while other DGP approaches do not converge as quickly as our technique.

Furthermore, the inclusion of an asynchronous implementation of stochastic gradient optimization allows us to exploit large-scale computing facilities using distributed updates, showing that we can truly scale DGPs to tractably tackle unprecedentedly large amounts of data.

To summarize, the contributions of this work are as follows: (i) we propose random Fourier feature approximations for DGPs; (ii) we discuss the connections between the proposed approximate DGP model and DNNs; (iii) we compare the quality of the proposed approximation/inference framework by visualizing the approximate functions at all layers against those obtained with MCMC techniques for a full two-layer DGP regression model; (iv) we demonstrate the ability of our proposal to outperform state-of-the-art methods to carry out inference in DGP models; (v) we describe and release a TensorFlow implementation of the proposed DGP model; (vi) we propose an asynchronous distributed version of the code that allows us to tackle unprecedented DGP modeling problems.

The paper is organized as follows. The rest of this section reports the related work and some background on GPs. Sec. 2 proposes random Fourier features for

DGPs, and shows how DGPs with squared exponential covariance functions are approximated by trigonometric DNNs that can be learned using stochastic variational inference. Sec. 3 reports an extensive evaluation of our proposal and a thorough comparison with state-of-the-art approaches to learn DGPs, while Sec. 4 concludes the paper.

1.1 Related work

Following the original proposal of DGP models in Damianou and Lawrence (2013), there have been several attempts to extend GP inference techniques to DGPs. Notable examples include the extension of inducing point approximations (Hensman and Lawrence, 2014; Dai et al., 2016) and Expectation Propagation (Bui et al., 2016). A recent example of a DGP “natively” formulated as a variational model appears in Tran et al. (2016). Our work is the first to employ random Fourier features to approximate DGPs as DNNs with trigonometric activation functions, whose properties were studied in Sopena et al. (1999).

The connection between DGPs and DNNs has been pointed out in several papers, such as Neal (1996) and Duvenaud et al. (2014), where pathologies with deep nets are investigated. The approximate DGP model described in our work becomes a DNN with low-rank weight matrices, which have been used in, e.g., Novikov et al. (2015); Sainath et al. (2013); Denil et al. (2013) as a regularization mechanism. Dropout is another technique to speed-up training and improve generalization of DNNs that has recently been linked to variational inference (Gal and Ghahramani, 2016).

Random Fourier features for large scale kernel machines were proposed in Rahimi and Recht (2008), and their application to GPs appears in Lázaro-Gredilla et al. (2010). Variational learning of the posterior over the frequencies was then proposed in Gal and Turner (2015) to avoid potential overfitting caused by optimizing these variables. These approaches are special cases of our DGP model when using no hidden layers.

In our work, we learn the proposed approximate DGP model using stochastic variational inference. Variational learning for DNNs was first proposed in Graves (2011), and later extended to include the so-called reparameterization trick to clamp randomness in the computation of the gradient with respect to the posterior over the weights (Kingma and Welling, 2014), and to include a Gaussian mixture prior over the weights (Blundell et al., 2015). Inference of Bayesian DNNs with MCMC was studied in Neal (1996).

1.2 Gaussian Processes

Consider a supervised learning scenario where a set of input vectors $X = [\mathbf{x}_1, \dots, \mathbf{x}_n]^\top$ is associated with a set of (possibly multivariate) labels $Y = [\mathbf{y}_1, \dots, \mathbf{y}_n]^\top$, where $\mathbf{x}_i \in \mathbb{R}^{D_{\text{in}}}$ and $\mathbf{y}_i \in \mathbb{R}^{D_{\text{out}}}$. We assume that there is an underlying function $f_o(\mathbf{x}_i)$ characterizing a mapping from the inputs to a latent (unobserved) representation, and that the labels are a realization of some probabilistic process $p(y_{io}|f_o(\mathbf{x}_i))$ which is based on this latent representation.

GP models assume a nonparametric form for the latent functions. Formally, GPs are a collection of random variables such that any subset of these are jointly Gaussian distributed (Rasmussen and Williams, 2006). In GPs, the covariance between variables at different locations, say $f_o(\mathbf{x}_i)$ and $f_o(\mathbf{x}_j)$, is modeled using a covariance function $k(\mathbf{x}_i, \mathbf{x}_j|\boldsymbol{\theta})$ parameterized via a set of parameters $\boldsymbol{\theta}$. A typical example of a covariance function, which we use throughout the paper, is the Radial Basis Function (RBF) covariance with Automatic Relevance Determination (ARD) (Mackay, 1994)

$$k(\mathbf{x}, \mathbf{x}'|\boldsymbol{\theta}) = \sigma^2 \exp \left[-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^\top \Lambda^{-1}(\mathbf{x} - \mathbf{x}') \right]. \quad (1)$$

The parameter set $\boldsymbol{\theta}$ comprises the marginal variance of the GP σ^2 , and $\Lambda = \text{diag}(l_1^2, \dots, l_d^2)$, with the l_i 's interpreted as lengthscale parameters along each of the dimensions of the input domain.

Denote by F the set of latent variables with entries $f_{io} = f_o(\mathbf{x}_i)$, and let $p(Y|F)$ be the conditional likelihood. Given a dataset of n observations, we aim to optimize or infer covariance parameters that requires repeated evaluation of the marginal likelihood $p(Y|X, \boldsymbol{\theta})$. After that, we wish to carry out predictions for a new test point \mathbf{x}_* by evaluating $p(\mathbf{y}_*|Y, X, \mathbf{x}_*, \boldsymbol{\theta})$. There are two main difficulties in dealing with these tasks. First, in the case of a non-Gaussian likelihood $p(Y|F)$, the marginal likelihood and the predictive probability are analytically intractable. Secondly, even in the case of a Gaussian likelihood where these are computable, the GP prior introduces the need to solve algebraic operations with a computational cost of $O(n^3)$. This poses a scalability issue for GPs that requires resorting to approximations.

2 DGPs WITH RANDOM FOURIER FEATURES

For the sake of clarity, we report the conventions used in the following sections. Variables in layer l are denoted by the (l) superscript. We assume N_h layers, each composed of a $D_{F^{(l)}}$ dimensional multivariate GP, as depicted in Figure 1.

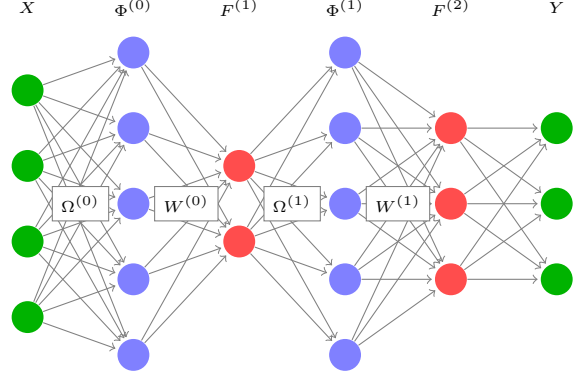


Figure 1: Diagram of the proposed DGP with random Fourier features.

2.1 Deep Gaussian Processes

In Deep Gaussian Processes (DGPs; Damianou and Lawrence, 2013), the mapping between inputs and labels is expressed as a composition of functions

$$\mathbf{f}(\mathbf{x}) = \left(\mathbf{h}^{(N_h-1)} \left(\boldsymbol{\theta}^{(N_h-1)} \right) \circ \dots \circ \mathbf{h}^{(0)} \left(\boldsymbol{\theta}^{(0)} \right) \right) (\mathbf{x}),$$

where each of the functions $\mathbf{h}^{(l)}(\boldsymbol{\theta}^{(l)})$ are modeled using (possibly transformed) multivariate Gaussian processes. Given that GPs are single layered neural networks with an infinite number of hidden units (Neal, 1996), the DGP model has an obvious connection with Deep Neural Networks (DNNs). In contrast to DNNs, where each of the hidden layers implements a function $\mathbf{h}^{(l)}(\boldsymbol{\theta}^{(l)})$ that is a transformation of linear combinations of its inputs, in DGPs these functions are assigned a GP prior, and are therefore nonparametric. This has a considerable advantage over DNNs in that the architecture of the DGP does not require tedious cross-validation to find the best configuration of number of neurons for each layer. Furthermore, by virtue of the probabilistic formulation of DGPs, it is in principle possible to use the model evidence to determine the optimal number of hidden layers.

While DGPs are attractive from a theoretical standpoint, inference is extremely challenging. Learning DGPs requires computing the probability of the labels given the input and all covariance parameters $p(Y|X, \boldsymbol{\theta})$. The integrals involved to evaluate this expression are generally intractable for any interesting nonlinear covariance functions.

2.2 DGPs with random Fourier features become DNNs

In this section, we present our approximate formulation of DGPs that, as we illustrate in the experiments, leads to a practical learning algorithm for these deep

probabilistic models. Appealing to Bochner’s theorem, any continuous shift-invariant normalized covariance function $k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_i - \mathbf{x}_j)$ is positive definite if and only if it can be rewritten as the Fourier transform of a non-negative measure $p(\boldsymbol{\omega})$ (Rahimi and Recht, 2008). Denoting the spectral frequencies by $\boldsymbol{\omega}$, while assigning $\iota = \sqrt{-1}$ and $\boldsymbol{\delta} = \mathbf{x}_i - \mathbf{x}_j$, in the case of the RBF covariance detailed in equation 1, this yields:

$$k(\boldsymbol{\delta}|\boldsymbol{\theta}) = \sigma^2 \int p(\boldsymbol{\omega}) \exp(\iota \boldsymbol{\delta}^\top \boldsymbol{\omega}) d\boldsymbol{\omega}, \quad (2)$$

with a corresponding non-negative measure $p(\boldsymbol{\omega}) = \mathcal{N}(\mathbf{0}, \Lambda^{-1})$. Because the covariance function and the non-negative measures are real, we can drop the unnecessary complex part of the argument of the expectation, keeping $\cos(\boldsymbol{\delta}^\top \boldsymbol{\omega}) = \cos((\mathbf{x}_i - \mathbf{x}_j)^\top \boldsymbol{\omega})$ that can be rewritten as $\cos(\mathbf{x}_i^\top \boldsymbol{\omega}) \cos(\mathbf{x}_j^\top \boldsymbol{\omega}) + \sin(\mathbf{x}_i^\top \boldsymbol{\omega}) \sin(\mathbf{x}_j^\top \boldsymbol{\omega})$.

The importance of the expansion above is that it allows us to interpret the covariance function as an expectation that can be estimated using Monte Carlo. Defining $\mathbf{z}(\mathbf{x}|\boldsymbol{\omega}) = [\cos(\mathbf{x}^\top \boldsymbol{\omega}), \sin(\mathbf{x}^\top \boldsymbol{\omega})]^\top$, the covariance function can be approximated by the following (unbiased) Monte Carlo estimate

$$k(\mathbf{x}_i, \mathbf{x}_j|\boldsymbol{\theta}) \approx \frac{\sigma^2}{N_{\text{RFF}}} \sum_{r=1}^{N_{\text{RFF}}} \mathbf{z}(\mathbf{x}_i|\tilde{\boldsymbol{\omega}}_r)^\top \mathbf{z}(\mathbf{x}_j|\tilde{\boldsymbol{\omega}}_r), \quad (3)$$

with $\tilde{\boldsymbol{\omega}}_r \sim p(\boldsymbol{\omega})$.

This has an important practical implication, as it provides the means to access an approximate explicit representation of the mapping induced by the covariance function that, in the RBF case, we know is infinite dimensional (Shawe-Taylor and Cristianini, 2004). Various results have been established on the accuracy of the random Fourier feature approximation; see e.g. Rahimi and Recht (2008)).

We propose to employ this approximation at each layer. Assume that the GPs have zero mean, and define $F^{(0)} := X$. By taking a “weight-space view” of the GPs at each layer, we have that

$$\Phi^{(l)} = \sqrt{\frac{\sigma^2}{N_{\text{RFF}}^{(l)}}} \left[\cos\left(F^{(l)} \Omega^{(l)}\right), \sin\left(F^{(l)} \Omega^{(l)}\right) \right], \quad (4)$$

and

$$F^{(l+1)} = \Phi^{(l)} W^{(l)}. \quad (5)$$

At each layer, the priors over the weights are $p(\Omega_j^{(l)}) = \mathcal{N}(\mathbf{0}, (\Lambda^{(l)})^{-1})$ and $p(W_i^{(l)}) = \mathcal{N}(\mathbf{0}, I)$.

Each matrix $\Omega^{(l)}$ is defined as a matrix with dimensions $D_{F^{(l)}} \times N_{\text{RFF}}^{(l)}$. On the other hand, the weight matrices $W^{(l)}$ are defined as matrices with dimensions $2N_{\text{RFF}}^{(l)} \times D_{F^{(l+1)}}$, with the constraint that $D_{F^{(N_h)}} =$

D_{out} . The accuracy of the approximation of these transformations with respect to the actual GPs is controlled by the parameters $N_{\text{RFF}}^{(l)}$, that determine how many random Fourier features are used to approximate the GPs at each layer.

Our formulation of an approximate DGP using random Fourier features reveals a close connection with DNNs. In our formulation, the design matrices at each layer are $\Phi^{(l+1)} = \gamma(\Phi^{(l)} W^{(l)} \Omega^{(l+1)})$, where $\gamma(\cdot)$ denotes the element-wise application of sine and cosine. In the DNN case, instead, the design matrices are computed as $\Phi^{(l+1)} = g(\Phi^{(l)} \Omega^{(l)})$, where $g(\cdot)$ is an activation function (e.g., sigmoid, ReLU). In light of this, we can view our approximate DGP model as a DNN. In particular, the resulting model is a Trigonometric DNN (Sopena et al., 1999) interlayered with linear transformations. From a probabilistic standpoint, we can interpret our approximate DGP model as a DNN with specific Gaussian priors over the $\Omega^{(l)}$ weights controlled by the covariance parameters, and Gaussian priors over the $W^{(l)}$ weights with unit variance. Covariance parameters act as some sort of hyper-priors over the covariance of $\Omega^{(l)}$, and the objective is to optimize these during training.

Another observation about the resulting DGP approximation is that, for a given layer l , the transformations given by $W^{(l)}$ and $\Omega^{(l+1)}$ are both linear. If we collapsed the two transformations into a single one, by introducing weights $\Xi^{(l)} = W^{(l)} \Omega^{(l+1)}$, we would have to learn $2N_{\text{RFF}}^{(l)} \times N_{\text{RFF}}^{(l+1)}$ weights at each layer, which is considerably more than learning the two separate sets of weights. As a result, we can view the proposed approximate DGP model as a way to impose a low-rank structure on the weights of DNNs, which is a form of regularization that has been proposed in the literature of DNNs (Novikov et al., 2015; Sainath et al., 2013; Denil et al., 2013).

2.3 Variational inference

In order to keep the notation uncluttered, let Ψ be the collection of all weight matrices $\Omega^{(l)}$ and $W^{(l)}$ at all layers, and $\boldsymbol{\theta}$ the collection of all covariance parameters $\boldsymbol{\theta}^{(l)}$ at all layers. Subsequently, given the following prior:

$$p(\Psi|\boldsymbol{\theta}) = \prod_{l=0}^{N_h-1} p(\Omega^{(l)}|\boldsymbol{\theta}^{(l)}) p(W^{(l)}), \quad (6)$$

we train the proposed approximate DGP using variational inference following Kingma and Welling (2014), and Graves (2011).

Note that we have different options for how to treat model parameters. We could fix Ω by drawing from

the prior induced by the covariance expansion and be variational about W . Alternatively, we could also be variational about Ω as well, which was proposed for shallow GPs in Gal and Turner (2015). We report the general formulation here, but in the appendix we report a derivation of the case where Ω is fixed. We also note that one could be variational about θ , but leave this for future work.

We are interested in computing $p(Y|\theta)$, as this would represent an objective function to learn the parameters θ . Computing $p(Y|\theta)$ involves intractable integrals, so we aim for an approximation. Defining $\mathcal{L} = \log[p(Y|\theta)]$, we obtain a lower bound using variational techniques as follows:

$$\mathcal{L} \geq \mathbb{E}_{q(\Psi)}(\log[p(Y|\Psi)]) - \text{DKL}[q(\Psi)||p(\Psi|\theta)], \quad (7)$$

where $q(\Psi)$ acts as an approximation to the posterior over all the weights $p(\Psi|Y, \theta)$.

We are interested in optimizing $q(\Psi)$, i.e. finding an optimal approximate distribution over the parameters according to the bound above. The first term can be interpreted as a model fitting term, whereas the second as a regularization term. In the case of a Gaussian distribution $q(\Psi)$ and a Gaussian prior $p(\Psi|\theta)$, it is possible to compute the DKL term analytically (see supplementary material), whereas the remaining term needs to be estimated. Assuming that the approximating distribution factorizes across layers, spectral frequencies and weights:

$$q(\Psi) = \prod_{ijl} q(\Omega_{ij}^{(l)}) \prod_{ijl} q(W_{ij}^{(l)}). \quad (8)$$

The variational parameters then become the mean and the variance of each of the approximating factors

$$q(W_{ij}^{(l)}) = \mathcal{N}(\mu_{ij}^{(l)}, (\sigma^2)_{ij}^{(l)}), \quad (9)$$

$$q(\Omega_{ij}^{(l)}) = \mathcal{N}(m_{ij}^{(l)}, (s^2)_{ij}^{(l)}), \quad (10)$$

and we aim to optimize the lower bound with respect to the variational parameters $\mu_{ij}^{(l)}, (\sigma^2)_{ij}^{(l)}, m_{ij}^{(l)}, (s^2)_{ij}^{(l)}$.

In the case of a likelihood that factorizes across observations, an interesting feature of the expression of the lower bound above is that it is amenable to stochastic optimization. In particular,

$$\mathbb{E}_{q(\Psi)}\left(\log\left[\prod_i p(y_k|\Psi)\right]\right) = \sum_k \mathbb{E}_{q(\Psi)}(\log[p(y_k|\Psi)]),$$

where each term $\mathbb{E}_{q(\Psi)}(\log[p(y_k|\Psi)])$ requires to be estimated, and we can do this using the Monte Carlo estimator

$$\mathbb{E}_{q(\Psi)}(\log[p(y_k|\Psi)]) \approx \frac{1}{N_{\text{MC}}} \sum_{r=1}^{N_{\text{MC}}} \log[p(y_k|\tilde{\Psi}_r)], \quad (11)$$

with $\tilde{\Psi}_r \sim q(\Psi)$. In order to facilitate the optimization, we employ a reparameterization of the weight matrices as follows:

$$(\tilde{W}_r^{(l)})_{ij} = \sigma_{ij}^{(l)} \varepsilon_{rij}^{(l)} + \mu_{ij}^{(l)}, \quad (12)$$

and

$$(\tilde{\Omega}_r^{(l)})_{ij} = s_{ij}^{(l)} \epsilon_{rij}^{(l)} + m_{ij}^{(l)}, \quad (13)$$

with $\varepsilon_{rij}^{(l)} \sim \mathcal{N}(0, 1)$ and $\epsilon_{rij}^{(l)} \sim \mathcal{N}(0, 1)$. In this way, at each iteration of the optimization over the parameters, we can fix the randomness in the computation of the expectation, and apply stochastic gradient ascent moves to the mean and variance of the approximating distribution over model parameters (Kingma and Welling, 2014).

We can also incorporate mini-batch learning by considering a mini-batch of data points having size m , and obtain an unbiased estimate of the lower bound as follows:

$$\frac{n}{m} \sum_{k \in \Omega} \mathbb{E}_{q(\Psi)}(\log[p(y_k|\Psi)]) - \text{DKL}[q(\Psi)||p(\Psi|\theta)]. \quad (14)$$

By differentiating this expression wrt mean and variance of the approximate posterior over Ψ , we obtain an unbiased estimate of the gradient of the lower bound. We can therefore apply stochastic gradient optimization to learn the optimal variational distribution $q(\Psi)$. Automatic differentiation tools allow us to carry out the computation of the stochastic gradient automatically, which is why we opt to implement our model in TensorFlow (Abadi et al., 2015).

2.4 Comparison with MCMC

Figure 2 shows a comparison between the variational approximation and MCMC for a two-layer DGP model applied to a regression dataset. The dataset has been generated by drawing 50 data points from $\mathcal{N}(y|h(h(x)), 0.01)$, with $h(x) = 2x \exp(-x^2)$. We experiment with two different levels of precision in the DGP approximation by using 10 and 50 spectral frequencies respectively, so as to assess the impact of this parameter on the results. For MCMC, covariance parameters are fixed to the values obtained by optimizing the variational lower bound on the marginal likelihood in the case of 50 spectral frequencies.

We obtained samples from the posterior over the latent variables at each layer using MCMC techniques. In the case of a Gaussian likelihood, it is possible to integrate out the GP at the last layer, thus obtaining a model that only depends on the GP at the first. As a result, the collapsed DGP model becomes a standard GP model whose latent variables can be sampled using various MCMC samplers developed in the literature of MCMC for GPs. Here we employ Elliptical

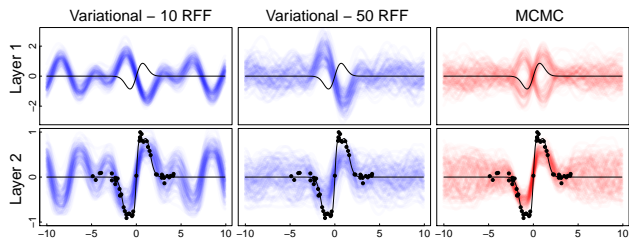


Figure 2: Comparison of MCMC and variational inference of a two-layer DGP with a single GP in the hidden layer and a Gaussian likelihood. The posterior over the latent functions is based on 100 MCMC samples and 100 samples from the variational posterior.

Slice Sampling (Murray et al., 2010) to draw samples from the posterior over the latent variables at the first layer, whereas latent variables at the second can be sampled directly from a multivariate Gaussian distribution. More details on the MCMC sampler may be found in the supplementary material.

The plots depicted in the figure illustrate how the MCMC approach explores two modes of the posterior of opposite sign. This is due to the invariance stemming from the possibility to obtain the same functions at the last layer by simply flipping the sign of the weight matrices at the two layers. Conversely, the variational approximation accurately identifies one of the two modes of the posterior. The overall approximation is accurate in the case of more random Fourier features, whereas in the case of less, the approximation is unsurprisingly characterized by out-of-sample oscillations. The variational approximation seems to result in larger uncertainty in predictions compared to MCMC; we attribute this to the factorization of the posterior over all the weights.

3 EXPERIMENTS

The main motivation for this work is the potential to train DGP models in an efficient and scalable manner without compromising on predictive performance. In view of these criteria, we thoroughly evaluate our model by comparing it against other recent DGP methods for both regression and classification, and assess its performance when applied to particularly complex and large datasets. We also experiment with an asynchronous version of the model in a distributed environment to enable the model to properly scale to large datasets.

3.1 Model Comparison

We compare our model to three of the most recent DGP methods presented in the literature,

namely DGPs using approximate expectation propagation (DGP-EP; Bui et al., 2016), variational auto-encoded DGPs (DGP-VAR; Dai et al., 2016), and the standard DGP by Damianou and Lawrence (2013). We use the same experimental set-up for both regression and classification tasks using datasets from the UCI repository (Asuncion and Newman, 2007), and repeat the experiments using two and three hidden layers, respectively. The specific configurations for each model are detailed below:

DGP-RFF : In proposed DGP with random Fourier features, we use 100 spectral frequencies at every hidden layer, and set the dimensionality to 3. So as to avoid large variance in our results, we set the number of Monte Carlo samples to 100 for both the training and test scenarios, and the batch size to 100. We employ the Adam optimizer provided in TensorFlow with a learning rate of 0.01, and in order to stabilize the optimization procedure, we fix the hyperparameters and the spectral frequencies for 4,000 iterations before jointly optimizing all parameters;

DGP-EP¹: For this technique, we use a batch size of 100 and 100 inducing points at each hidden layer, and we also set the dimensionality of the hidden layer to 3. The Adam optimizer is initialized with the standard learning rate, although this is then adapted within the model itself during optimization. For the classification case, we use 100 samples for approximating the Softmax likelihood;

DGP-VAR²: As above, 100 inducing points and a dimensionality of 3 is employed at every hidden layer. For the multi-layer perceptron in the recognition model, we use the provided standard settings. Feed-forwarding of the original inputs is disabled;

DGP²: The set-up of this model is identical to that of **DGP-VAR**, with the exception that the recognition model is absent.

We assess the performance of each model by running the optimization procedure for a set period of time, and periodically evaluate the error rate and mean negative log-likelihood (MNLL) on withheld test data. The results are averaged over 3 folds for every dataset. The experiments were launched on a cluster of servers with Intel Xeon E5-2630 CPUs having 32 cores and 128GB RAM, while ensuring that the runs for all methods are allocated a fair amount of resources.

¹Code obtained from:
github.com/thangbui/deepGP_approxEP

²Code obtained from:
github.com/zhenwendai/DeepGP/tree/master/deepgp

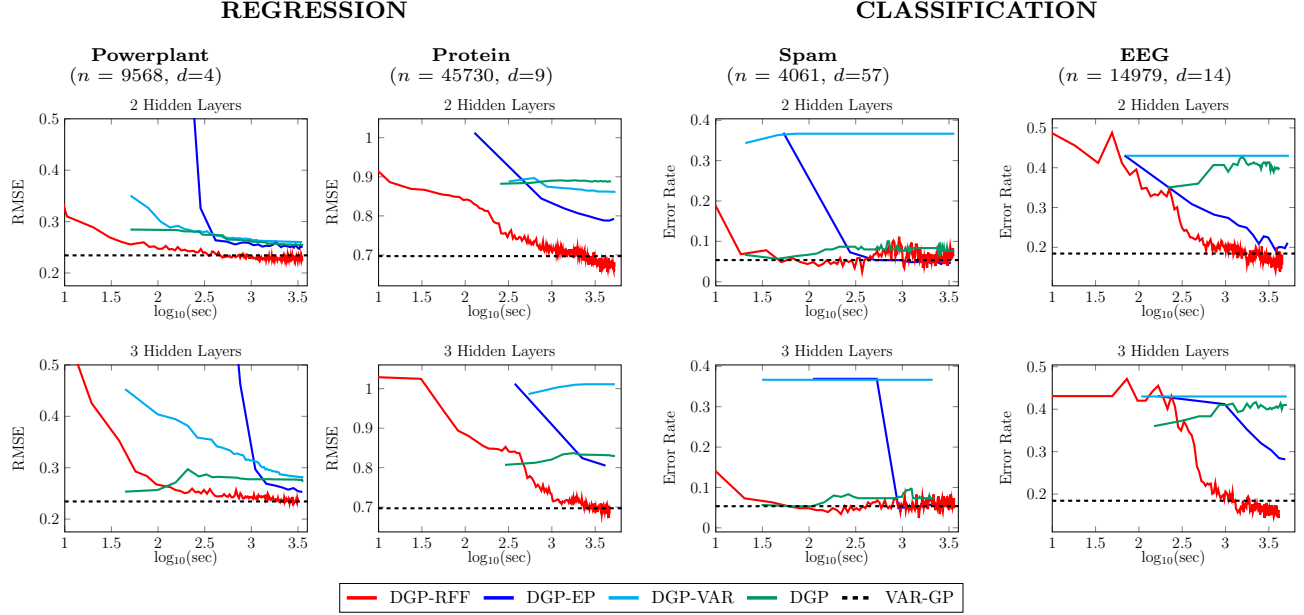


Figure 3: Progression of RMSE over time for competing DGP models over two datasets for regression (Powerplant and Protein), and two for classification (Spam and EEG). Results are shown for configurations having 1 and 2 hidden layers, respectively. Dashed line indicates the accuracy obtained by a Variational GP (Titsias, 2009).

The results illustrated in Figure 3 demonstrate that DGP-RFF outperforms other DGP techniques both in terms of convergence speed and overall predictive accuracy. This is particularly significant for larger datasets where other techniques take considerably longer to converge to a reasonable error rate. It is also of note that the model performs well even when two hidden layers are considered, whereas the competing techniques do not adapt as well. The results are also competitive (and sometimes superior) to those obtained by the variational GP in Titsias (2009).

Whereas the performance of DGP-EP seems to follow a similar (albeit slower) trajectory to our method, the results obtained for DGP-VAR and the standard DGP methods are notably more erratic. This can be partly attributed to the fact that the available implementation of this model caters mostly for Gaussian likelihoods, which are known to be poorly suited to classification tasks. Even so, the performance of these two models on the Protein dataset is also inferior, and altering the default parameter configurations did not seem to make much of a difference.

As highlighted in the introduction to this work, one of the most appealing features of DGPs over DNNs is the ability to quantify the uncertainty of predictions. In Table 1, we tabulate the MNLL obtained by DGP-RFF and its closest competitor, DGP-EP, after optimization is terminated. DGP-RFF performs better in all but one dataset, which happens to have the greatest dimensionality (57 features). This reveals that

more spectral frequencies may be required in large-dimensional cases.

Table 1: Comparison of MNLL for DGP-RFF and DGP-EP using 1 hidden layer with fixed time budget.

DATASET	DGP-EP	DGP-RFF
Powerplant	0.059	-0.067
Protein	1.062	1.032
Spam	0.137	0.193
EEG	0.415	0.374

3.2 Multi-class Classification

As a demonstrative example, we evaluate the model using the standard 10-class MNIST dataset, which involves classifying a database of hand-written digits.

In Table 2, we frame our results in the context of alternative learning models, and show how different configurations of DGP-RFF perform on this dataset. In particular, we evaluate our model using no hidden layers (whereby it is reduced to a standard variational GP), and also using 1 and 2 hidden layers. Across all variations, we use 500 spectral frequencies, 40 GPs in the hidden layers, Adam with standard learning rate, and fix the hyperparameters and the spectral frequencies for 4,000 iterations.

The results indicate that using more hidden layers does not necessarily imply superior performance, and al-

though the results for the DGP-RFF variations presented here plateau around the 98.2% region, the model performance degrades noticeably when more than 2 hidden layers are used. This is in line with what is reported in the literature on DNNs (Neal, 1996; Duvenaud et al., 2014). Nonetheless, especially when considering previous results obtained by GP models, the predictive accuracy achieved by this model is very competitive to other well-known learning techniques.

Table 2: Comparison of accuracy for the proposed DGP-RFF approach with previous results on MNIST. The comparison focuses on approaches without pre-processing, and excludes convolutional neural nets.

ALGORITHM	ACC.
Sparse GP-LVM (Gal et al., 2014)	94.00%
GP Hashing (Ozdemir and Davis, 2016)	$\sim 95.00\%$
MCMC VAR-GP (Hensman et al., 2015)	98.04%
DNN (Simard et al., 2003)	98.50%
SVM (Schölkopf, 1997)	98.60%
DGP-RFF 0HL	98.36%
DGP-RFF 1HL	98.10%
DGP-RFF 2HL	97.44%

We also evaluated the performance of our model on MNIST-8M, which artificially extends the original MNIST dataset to over 8 million observations, and we obtained 99.11% accuracy on the test set using 1 hidden layer. This goes far beyond the scale of datasets to which Gaussian processes (and their Deep variations) are usually applied, giving further credence to the tractability and robustness of our model.

3.3 Distributed Implementation

Our model is easily amenable to a distributed implementation using asynchronous distributed stochastic gradient descent (Chilimbi et al., 2014; Abadi et al., 2015). Our distributed setting³, based on TensorFlow, includes one or more *Parameter servers* (PS), and a number of *Workers*. The latter proceed asynchronously using randomly selected batches of data: they fetch fresh model parameters from the PS, compute the gradients of the loss function with respect to these parameters, and push those gradients back to the PS, which update the model accordingly. Given that workers compute gradients and send updates to PS asynchronously, the discrepancy between the model used to compute gradients and the model actually updated can degrade training quality. This is exacerbated by a large number of asynchronous workers, as

³Currently, we use a CPU-only compute cluster.

noted in Chen et al. (2016).

We focus our experiments on the MNIST dataset, and study how training time and error rates evolve with the number of workers introduced in our system. The parameters for the model are identical to those reported for the previous experiment.

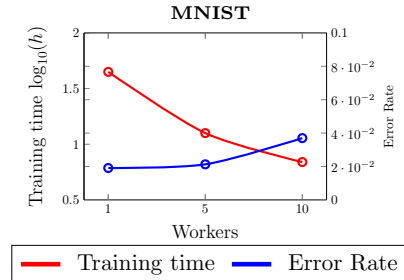


Figure 4: Comparison of training time and error rate for asynchronous DGP-RFF with 1, 5 and 10 workers.

We report the results in Figure 4, and as expected, the training time decreases in proportion to the number of workers, albeit sub-linearly. On the other hand, the increasing error rate confirms our intuition that imprecise updates of the gradients negatively impact the optimization procedure. The work in Chen et al. (2016) corroborates our findings.

4 CONCLUSIONS

In this work, we have proposed a novel formulation of DGPs which exploits the approximation of the kernel function using random Fourier features, as well as stochastic variational inference for preserving the probabilistic representation of a regular GP. As evidenced by the results obtained in our experiments, our approach works noticeably better than recent state-of-the-art DGP models for both regression and classification tasks. Moreover, we obtained highly competitive results for both the MNIST and MNIST-8M digit recognition problems, the latter of which has been generally considered to be beyond the computational scope of GPs. The asynchronous implementation of the model also permits us to scale to even larger datasets than those considered in this paper.

We are currently investigating ways to mitigate the oscillatory behavior observed in our comparison with MCMC sampling, and architectures where the original input is connected to all layers, which could potentially improve performance (Duvenaud et al., 2014). The obtained results also encourage the extension of DGP-RFF to include convolutional layers suitable for computer vision applications. The implementation of a distributed synchronous version is also in the pipeline.

References

- M. Abadi, A. Agarwal, P. Barham, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.
- A. Asuncion and D. J. Newman. UCI machine learning repository, 2007.
- C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight Uncertainty in Neural Network. In F. R. Bach and D. M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1613–1622. JMLR.org, 2015.
- T. D. Bui, D. Hernández-Lobato, J. M. Hernández-Lobato, Y. Li, and R. E. Turner. Deep Gaussian Processes for Regression using Approximate Expectation Propagation. In M. Balcan and K. Q. Weinberger, editors, *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1472–1481. JMLR.org, 2016.
- J. Chen, R. Monga, S. Bengio, and R. Jozefowicz. Revisiting distributed synchronous sgd. <https://arxiv.org/abs/1604.00981>, 2016.
- T. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman. Project adam: Building an efficient and scalable deep learning training system. In *USENIX Symposium on Operating Systems Design and Implementation, October 6-8, 2014, Broomfield, Colorado, USA*, 2014.
- Z. Dai, A. Damianou, J. González, and N. Lawrence. Variational Auto-encoded Deep Gaussian Processes, Feb. 2016.
- A. C. Damianou and N. D. Lawrence. Deep Gaussian Processes. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, AZ, USA, April 29 - May 1, 2013*, volume 31 of *JMLR Proceedings*, pages 207–215. JMLR.org, 2013.
- M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. de Freitas. Predicting Parameters in Deep Learning. In C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 2148–2156, 2013.
- D. K. Duvenaud, O. Rippel, R. P. Adams, and Z. Ghahramani. Avoiding pathologies in very deep networks. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*, volume 33 of *JMLR Workshop and Conference Proceedings*, pages 202–210. JMLR.org, 2014.
- Y. Gal and Z. Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In M. Balcan and K. Q. Weinberger, editors, *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1050–1059. JMLR.org, 2016.
- Y. Gal and R. Turner. Improving the Gaussian Process Sparse Spectrum Approximation by Representing Uncertainty in Frequency Inputs. In F. R. Bach and D. M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 655–664. JMLR.org, 2015.
- Y. Gal, M. van der Wilk, and C. E. Rasmussen. Distributed Variational Inference in Sparse Gaussian Process Regression and Latent Variable Models. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3257–3265, 2014.
- A. Graves. Practical Variational Inference for Neural Networks. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2348–2356. Curran Associates, Inc., 2011.
- J. Hensman and N. D. Lawrence. Nested Variational Compression in Deep Gaussian Processes, Dec. 2014.
- J. Hensman, A. G. de G. Matthews, M. Filippone, and Z. Ghahramani. MCMC for variationally sparse gaussian processes. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1648–1656, 2015.
- D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *Proceedings of the Second International Conference on Learning Representations (ICLR 2014)*, Apr. 2014.
- M. Lázaro-Gredilla, J. Quinonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal. Sparse Spectrum Gaussian Process Regression. *Journal of Machine Learning Research*, 11:1865–1881, 2010.

- Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- D. J. C. Mackay. Bayesian methods for backpropagation networks. In E. Domany, J. L. van Hemmen, and K. Schulten, editors, *Models of Neural Networks III*, chapter 6, pages 211–254. Springer, 1994.
- I. Murray, R. P. Adams, and D. J. C. MacKay. Elliptical slice sampling. *Journal of Machine Learning Research - Proceedings Track*, 9:541–548, 2010.
- R. M. Neal. *Bayesian Learning for Neural Networks (Lecture Notes in Statistics)*. Springer, 1 edition, Aug. 1996. ISBN 0387947248.
- A. Novikov, D. Podoprikin, A. Osokin, and D. P. Vetrov. Tensorizing Neural Networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 442–450, 2015.
- B. Ozdemir and L. S. Davis. Scalable Gaussian Processes for Supervised Hashing, Apr. 2016.
- A. Rahimi and B. Recht. Random Features for Large-Scale Kernel Machines. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1177–1184. Curran Associates, Inc., 2008.
- C. E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.
- T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran. Low-rank matrix factorization for Deep Neural Network training with high-dimensional output targets. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, pages 6655–6659. IEEE, 2013. doi: 10.1109/ICASSP.2013.6638949.
- B. Schölkopf. *Support vector learning*. PhD thesis, Berlin Institute of Technology, 1997.
- J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.
- P. Y. Simard, D. Steinkraus, and J. C. Platt. Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2, ICDAR '03*, Washington, DC, USA, 2003. IEEE Computer Society.
- J. M. Sopena, E. Romero, and R. Alquezar. Neural networks with periodic and monotonic activation functions: a comparative study in classification problems. In *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, volume 1, 1999. doi: 10.1049/cp:19991129.
- M. K. Titsias. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In D. A. Dyk and M. Welling, editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009*, volume 5 of *JMLR Proceedings*, pages 567–574. JMLR.org, 2009.
- D. Tran, R. Ranganath, and D. M. Blei. The Variational Gaussian Process, Apr. 2016.

A Derivation of the lower bound

For the sake of completeness, here is a detailed derivation of the lower bound that we use in variational inference to learn the posterior over all weights in the model, i.e. both the spectral frequencies Ω and the weights W . Taking $\Psi = \{W, \Omega\}$:

$$\begin{aligned}
 \log [p(Y|\boldsymbol{\theta})] &= \log \left[\int p(Y|\Psi) p(\Psi|\boldsymbol{\theta}) d\Psi \right] \\
 &= \log \left[\int \frac{p(Y|\Psi) p(\Psi|\boldsymbol{\theta})}{q(\Psi)} q(\Psi) d\Psi \right] \\
 &= \log \left[\mathbb{E}_{q(\Psi)} \frac{p(Y|\Psi) p(\Psi|\boldsymbol{\theta})}{q(\Psi)} \right] \\
 &\geq \mathbb{E}_{q(\Psi)} \left(\log \left[\frac{p(Y|\Psi) p(\Psi|\boldsymbol{\theta})}{q(\Psi)} \right] \right) \\
 &= \mathbb{E}_{q(\Psi)} (\log [p(Y|\Psi)]) + \mathbb{E}_{q(\Psi)} \left(\log \left[\frac{p(\Psi|\boldsymbol{\theta})}{q(\Psi)} \right] \right) \\
 &= \mathbb{E}_{q(\Psi)} (\log [p(Y|\Psi)]) - \text{DKL}[q(\Psi) || p(\Psi|\boldsymbol{\theta})]
 \end{aligned}$$

Alternatively, we can also choose that the spectral frequencies, $\boldsymbol{\omega}$, are fixed. This results in:

$$\begin{aligned}
 \log [p(Y|\Omega)] &= \log \left[\int p(Y|W, \Omega) p(W) dW \right] \\
 &= \log \left[\int \frac{p(Y|W, \Omega) p(W)}{q(W)} q(W) dW \right] \\
 &= \log \left[\mathbb{E}_{q(W)} \frac{p(Y|W, \Omega) p(W)}{q(W)} \right] \\
 &\geq \mathbb{E}_{q(W)} \left(\log \left[\frac{p(Y|W, \Omega) p(W)}{q(W)} \right] \right) \\
 &= \mathbb{E}_{q(W)} (\log [p(Y|W, \Omega)]) + \mathbb{E}_{q(W)} \left(\log \left[\frac{p(W)}{q(W)} \right] \right) \\
 &= \mathbb{E}_{q(W)} (\log [p(Y|W, \Omega)]) - \text{DKL}[q(W) || p(W)]
 \end{aligned}$$

B Expression for the DKL divergence between Gaussians

Given $p_1(x) = \mathcal{N}(\mu_1, \sigma_1^2)$ and $p_2(x) = \mathcal{N}(\mu_2, \sigma_2^2)$, the KL divergence between the two is:

$$\text{DKL}(p_1(x) || p_2(x)) = \frac{1}{2} \left[\log \left(\frac{\sigma_2^2}{\sigma_1^2} \right) - 1 + \frac{\sigma_1^2}{\sigma_2^2} + \frac{(\mu_1 - \mu_2)^2}{\sigma_2^2} \right]$$

C Details of MCMC sampler for a two-layer DGP with a Gaussian likelihood

We give details of the MCMC sampler that we used to draw samples from the posterior over latent variables in DGPs. In the experiments, we regard this as the gold-standard against which we compare the quality of the proposed DGP approximation and inference. For the sake of tractability, we assume a two-layer DGP with a Gaussian likelihood, and we fix the hyper-parameters of the GPs. Without loss of generality, we assume Y to be univariate and the hidden layer to be composed of a single GP. The model is therefore as follows:

$$\begin{aligned}
 p(Y|F^{(2)}, \lambda) &= \mathcal{N}(Y|F^{(2)}, \lambda I) \\
 p(F^{(2)}|F^{(1)}, \boldsymbol{\theta}^{(1)}) &= \mathcal{N}(F^{(2)}|\mathbf{0}, K(F^{(1)}, \boldsymbol{\theta}^{(1)})) \\
 p(F^{(1)}|X, \boldsymbol{\theta}^{(0)}) &= \mathcal{N}(F^{(1)}|\mathbf{0}, K(X, \boldsymbol{\theta}^{(0)}))
 \end{aligned}$$

with λ , $\boldsymbol{\theta}^{(1)}$, and $\boldsymbol{\theta}^{(0)}$ fixed. In the model specification above, we denoted by $K(F^{(1)}, \boldsymbol{\theta}^{(1)})$ and $K(X, \boldsymbol{\theta}^{(0)})$ the covariance matrices obtained by applying the covariance function with parameters $\boldsymbol{\theta}^{(1)}$, and $\boldsymbol{\theta}^{(0)}$ to all pairs of $F^{(1)}$ and X , respectively.

Given that the likelihood is Gaussian, it is possible to integrate out $F^{(2)}$ analytically

$$p(Y|F^{(1)}, \lambda, \boldsymbol{\theta}^{(1)}) = \int p(Y|F^{(2)}, \lambda) p(F^{(2)}|F^{(1)}, \boldsymbol{\theta}^{(1)}) dF^{(2)}$$

obtaining the more compact model specification:

$$\begin{aligned} p(Y|F^{(1)}, \lambda, \boldsymbol{\theta}^{(1)}) &= \mathcal{N}(Y|\mathbf{0}, K(F^{(1)}, \boldsymbol{\theta}^{(1)}) + \lambda I) \\ p(F^{(1)}|X, \boldsymbol{\theta}^{(0)}) &= \mathcal{N}(F^{(1)}|\mathbf{0}, K(X, \boldsymbol{\theta}^{(0)})) \end{aligned}$$

For fixed hyper-parameters, these expressions reveal that the observations are distributed as in the standard GP regression case, with the only difference that the covariance is now parameterized by GP distributed random variables $F^{(1)}$. We can interpret these variables as some sort of hyper-parameters, and we can attempt to use standard MCMC methods to samples from their posterior.

In order to develop a sampler for all latent variables, we factorize their full posterior as follows:

$$p(F^{(2)}, F^{(1)}|Y, X, \lambda, \boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(0)}) = p(F^{(2)}|Y, F^{(1)}, \lambda, \boldsymbol{\theta}^{(1)}) p(F^{(1)}|Y, X, \lambda, \boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(0)})$$

which suggest a Gibbs sampling strategy to draw samples from the posterior where we iterate

1. sample from $p(F^{(1)}|Y, X, \lambda, \boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(0)})$
2. sample from $p(F^{(2)}|Y, F^{(1)}, \lambda, \boldsymbol{\theta}^{(1)})$

Step 1. can be done by setting up a Markov chain with invariant distribution given by:

$$p(F^{(1)}|Y, X, \lambda, \boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(0)}) \propto p(Y|F^{(1)}, \lambda, \boldsymbol{\theta}^{(1)}) p(F^{(1)}|X, \boldsymbol{\theta}^{(0)})$$

We can interpret this as a GP model, where the likelihood now assumes a complex form because of the nonlinear way in which the likelihood depends on $F^{(1)}$. Because of this interpretation, we can attempt to use any of the samplers developed in the literature of GPs to obtain samples from the posterior over latent variables in GP models.

Step 2. can be done directly given that the posterior over $F^{(2)}$ is available in closed form and it is Gaussian:

$$p(F^{(2)}|Y, F^{(1)}, \lambda, \boldsymbol{\theta}^{(1)}) = \mathcal{N}\left(F^{(2)} \middle| K^{(1)} (K^{(1)} + \lambda I)^{-1} Y, K^{(1)} - K^{(1)} (K^{(1)} + \lambda I)^{-1} K^{(1)}\right)$$

where we have defined

$$K^{(1)} := K(F^{(1)}, \boldsymbol{\theta}^{(1)})$$